Robot Coverage Control by Evolved Neuromodulation

Kyle I. Harrington, Emmanuel Awa, Sylvain Cussat-Blanc, Jordan Pollack

Abstract— **An important connection between evolution and learning was made over a century ago and is now termed as the Baldwin effect. Learning acts as a guide for an evolutionary search process. In this study reinforcement learning agents are trained to solve the robot coverage control problem. These agents are improved by evolving neuromodulatory gene regulatory networks (GRN) that influence the learning and memory of agents. Agents trained by these neuromodulatory GRNs can consistently generalize better than agents trained with fixed parameter settings. This work introduces evolutionary GRN models into the context of neuromodulation and illustrates some of the benefits that stem from neuromodulatory GRNs.**

I. Introduction

As biological organisms have evolved, there has been a general trend that appears to favor the selection of individuals with plasticity via learning. While basic hard-coded behaviors can be quite powerful and may even be complex, such behaviors do not often translate between environmental contexts. Furthermore, there is evidence from studies of the interaction between evolution and learning that, in tandem, both processes can exhibit a synergistic expediting effect [17], [3]. This has been termed the Baldwin expediting effect [4], which states that plasticity (learning) can facilitate genotype improvement by providing gradient information. We extend this research by evolving an artificial gene regulatory network (GRN), which dynamically modulates agent learning. In this study the neuromodulatory GRN allows agents to dynamically regulate learning and memory according to what the agent senses in its environment.

The interaction between evolution and learning has a long-standing history [17]. In their pioneering work, Hinton and Nowlan simultaneously evolve static and learnable neural network weights. They find that the ability to learn improves the evolvability of the networks. This improvement is expected to be primarily caused by the gradient information that learning provides about how far a network is from the correct solution. Since this initial finding there have been a number of studies which explore the Baldwin expediting effect [39], [3], [2], where a general consensus finds that the problem domain can have a significant impact on how much improvement, or even detriment, plasticity provides.

GRNs are a fundamental and pervasive structure in biology. Many motile bacterial cells use GRNs to modulate their flagellate motor allowing for chemotaxis, phototaxis, and other environment seeking behaviors. This property has been exploited for the control of robots with artificial GRNs [41], [27], [19], [10], where robots evolved useful navigation behaviors. Other research has extended the application of artificial GRNs to other robot control problems.

Artificial neural networks are one of the most used models of biologically-plausible learning because of their relation to biological neural networks. While the field of neural networks itself has had a tumultuous history, researchers continued to pursue studies of hierarchical learning structures, such as the Hinton and Nowlan's layering of generational evolution and lifetime learning. During the first neural network rennaisance a method for learning with cascading neural networks was proposed [30]. In a two-network cascade one network acts as a function network while the other network defines context by specifying the weights of the function network. In this work we consider an internal cascading mechanism, whereby a gene regulatory network produces neuromodulators that alter learning and memory.

Ackley and Littman designed a system to test learning and evolution with a modified backpropagation network in an artificial life context [1]. In their study agents evolve the initial weights of learnable-control neural networks and fixed reward-estimation networks. Fixed reward-estimation networks are used to produce a reinforcement signal for an adapted version of the backpropagation learning algorithm. The complexity of their environment and relatively open-ended task require such an estimated reinforcement signal. However, the robot coverage control problem which we address (explained in the following section) has a clear objective with a minimal natural framing in the context of reinforcement learning. This reduces the number of factors that may obscure analysis of the resultant behaviors.

We begin this paper with an overview of the robot coverage control problem. We then explain the reinforcement learning algorithm which is used for robot control. The gene regulatory networks that are used for neuromodulation are then introduced. Finally, we present the specific instantiation of the problem and the algorithms used in our experiments, and the experiments themselves. We then conclude with a discussion of the benefits of evolving neuromodulatory gene regulatory networks that influence learning and memory.

II. Coverage Control

Some of the classic examplar problems of artificial intelligence involve the control of agents in 2D worlds [32]. A high-impact problem in this domain is the control of robotic vacuum cleaners. The general robotic cleaner market is currently approaching the billion dollar mark. A robot vacuum cleaner is an agent that must avoid potentially mobile obstacles and clean ephemerally accumulating dirt within some region. Other tasks, such as charging are ignored in

K. Harrington, E. Awa, and J. Pollack are with the DEMO Lab, Department of Computer Science, Brandeis University, Waltham, MA, USA. S. Cussat-Blanc is with the University of Toulouse, Toulouse, France. Corresponding email: kyleh@cs.brandeis.edu (eawa@brandeis.edu, sylvain.cussat-blanc@irit.fr, pollack@brandeis.edu)

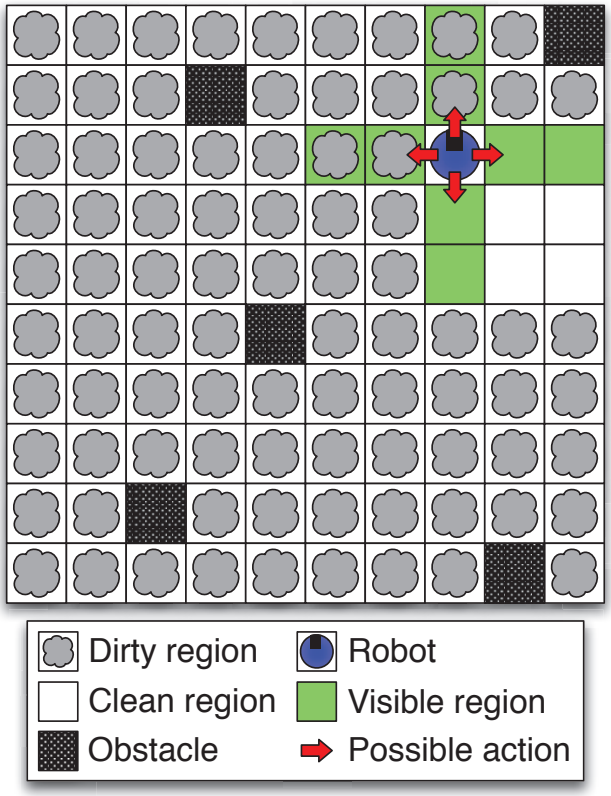


Fig. 1: The environment for the CCP problem.

this study. The objective of the robot is to clean all the dirt that has accumulated. In spite of limited sensors and the inability to maintain a sufficiently accurate internal map, this challenge of the task is to uniformly covering the region.

Robot coverage problems have a long standing history in evolutionary computation. One of the first genetic programming (GP) studies was on what has come to be called the "lawnmower" problem [12]. The lawnmower problem is an instance of the control coverage problem (CCP) in a uniform environment, often with turn-based navigation and the ability to jump. The use of modular structures have been shown to facilitate solving the lawnmower problem in GP [21]. The lawnmower problem, and other versions of the CCP, have been extensively studied in the field of GP, but a review of GP techniques is beyond the scope of this paper. However, we highlight an observation that was revealed for a version of the CCP with obstacles [35]. In this study the meanings of evolved behavioral modules were investigated, and it was found that optimal solutions contained modules that encoded short action sequences which were then iterated to solve the CCP. This observation suggests that successful agents may only need to learn a few short, but useful, action sequences. An example illustration of the CCP is shown in Figure 1.

The CCP has also been studied from a multi-robot perspective. In [26], simple hard-coded heuristics, such as wall-seeking behavior, are used by multiple robots to solve the CCP in a distributed manner. Optimal control for multi-robot coverage problems in mobile sensing networks has been explored analytically [8]. An overview of distributed algorithms for the CCP can be found in [7].

III. Reinforcement Learning

Reinforcement learning is a reward-based learning algorithm that allows agents to learn from experience. More formally, reinforcement learning (RL) is a mathematical framework for learning from a reward signal that is derived from Bellman's equation for optimal control [38]. One of the most important forms of RL is temporal-difference (TD) RL. TD-RL is a method for learning optimal behavior from changes in state and reinforcement by error prediction [37]. TD-RL agents learn an expected return that will be received after taking an action in any state. Strong correlations with this type of error predictive behavior have been found in studies of dopamine neurons [33]. This line of research has continued and is now been supported by fMRI data of reward processing for tastes, money, and love [16].

TD-RL is used to solve Markov decision processes, which are an extension of Markov chains to problems framed in terms of state, action, and reward. Reward signals (such as reinforcement of dirt cleanup) are geometrically encoded in a table which associates action preferences with states. The basic TD(γ) algorithm updates one state-action association at a time which prohibits sequence learning. Eligibility traces are used to associate reward with sequences of actions by reinforcing a weighted history of most recent actions. In this study the online version of TD-RL, SARSA (short for, state-action-reward-state-action), is used. A review of the nuances of reinforcement learning can be found in [38].

We include a few of the key equations from the RL algorithm which are employed. If we are in state, s_t at time t, then we will take some action a_t which will bring us a reward r_t. This action will also cause us to transition to a new state, s_{t+1}. The SARSA algorithm learns a Q-function, which maps a value to each state-action pair, (s_t, a_t). From each state multiple actions, A_t, may be taken which may be a function of s_t (for example, an obstacle may prevent an action in a given state). Given an optimal Q-function the best action to take is

$$argmin_{a_t \in A_t} Q(s_t, a_t). \tag{1}$$

The Q-function is approximated by SARSA with the following update rule

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\big[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\big] \tag{2}$$

where α is the learning rate, and γ is the discounting factor. Given only this update rule it can be difficult to compute the Q-value for state-action pairs which indirectly contribute to obtaining a reward. This update method propagates information only to the preceeding state-action pair, for those that are very distant from the reward, such as in the case of maze solving problems, this can require a large number of repeated trials. However, this problem of reward propagation can be

partially aleviated by the use of eligibility traces. Eligibility traces store an accumulating trace of state-action pairs. The "memory" of these state-action pairs can be tuned with the trace decay parameter λ. Eligibility traces are updated with

$$e_t(s,a) = \begin{cases} \gamma\lambda e_{t-1}(s,a) & \text{if } s \neq s_t \\ \gamma\lambda e_{t-1}(s,a) + 1 & \text{if } s = s_t \end{cases} \quad (3)$$

By combining the error predictive capabilities of TD-RL with the state-action sequence memory of eligibility traces we can amplify the effects of our reward and speed up the learning process. When performing on-policy learning it is important to ensure that a sufficient amount of exploration occurs. To this end the ϵ-greedy method is used, where a random action is taken with $p(\epsilon)$, otherwise the agent's most preferred action is taken. However, the RL algorithm can still fail to capitalize on rarely experienced rewards.

In this work, we propose to reduce the effect of rarely experienced rewards by dynamically modifying the RL parameters according to the local vision of the robot. With this aim in mind, we propose to use a gene regulatory network to supervise the adaptation of the coefficient. The end of this section presents the neuromodulation concept and the GRN model that we used as a neuromodulator.

IV. Neuromodulation

Neuromodulators are neuropeptides or small molecules, like dopamine and serotonin. The production of these substances within the cell is controlled by gene regulatory networks. Neuromodulators change the behavior of neural networks within individual neurons, amongst neighboring neurons, or throughout the entire network. Neuromodulation has been found to be pervasive throughout the brain, and can have drastic consequences on the behavior of neurons and neuronal circuits [11], [23], [24]. A particularly applicable example in the realm of robotics is the neuromodulation of motor signals produced by central pattern generators in the brain and spinal cord [20]. It has been found that neuromodulators tune and synchronize neuromuscular signals [40].

We have already noted that the temporal difference learning algorithm for error prediction has been observed in neural substrates [33]. Dopamine neurons of the ventral tegmental area (VTA) and substantia nigra exhibit this error predictive behavior. The dopamine system is itself a neuromodulatory system. While the temporal difference learning algorithm extends ideas of reward processing to engineering, there are models of the dopamine system with closer ties to biology [25]. These models also confirm the error predictive behavior found in the brain for a variety of physiological data including reaction-time and spatial-choice tasks. Dopamine is an important neuromodulator, especially in learning, but it is but one of many neuromodulatory substances found in the brain. An extensive review of computational models of neuromodulation can be found in [15], and some recent models are reviewed in [23]. In this study we focus on the relationship between evolved neuromodulator-producing GRNs and learned behaviors.

Computational studies of neuromodulation and neuroendocrine systems are becoming a popular method for acheiving dynamic control and learning. A number of these studies focus on neuromodulatory subsystems with projections that have diffuse action on the synapses of more classical neurons [36], [31], [34], [29]. On the other hand, this study develops a model of a single neuron (or homogenous nuclei) with dynamic regulation of learning and memory by an internal gene regulatory network. The idea of relating neuromodulatory substances to properties and learning parameters has been explored [14], [22], but to the best of our knowledge has not previously been evolved. As opposed to focusing on the spatial distribution of neuromodulatory action, our study focuses on how neuromodulation can be evolved to act in different ways on a learning system. In a related study, Benuskova and Kasabov evolve GRNs to tune the behavior of a biologically comprehensive spiking neural network [6]. In this study, we employ an abstract computational model of a gene regulatory network to modulate the parameters of a RL agent.

A. Gene regulatory network

Our model uses an optimized network of abstract proteins. The inputs of the agent are translated to protein concentrations that feed the GRN. Output proteins regulate the reinforcement learning parameters previously described. This kind of controller has been used in many developmental models of the literature [18], [13], [9] and to control virtual and real robots [41], [27], [19], [10].

We have based our regulatory network on Banzhaf's model [5]. It is designed to be as close as possible to a real gene regulatory network but neither to be evolved nor to control any kind of agent. However, Nicolau used an evolution strategy to evolve the GRN to control a pole-balancing cart [27]. Though this experiment behaved consistently, the evolution of the GRN has been an issue. We have decided to modify the encoding of the regulatory network and its dynamics. In our model, a gene regulatory network is defined as a set of proteins. Each protein has the following properties:

- The protein *tag* coded as an integer between 0 and p. The upper value p of the domain can be changed in order to control the precision of the GRN. In Banzhaf's work, p is equivalent to the size of a site, 16 bits.
- The *enhancer tag* coded as an integer between 0 and p. The enhancer tag is used to calculate the enhancing matching factor between two proteins.
- The *inhibitor tag* coded as an integer between 0 and p. The inhibitor tag is used to calculate the inhibiting matching factor between two proteins.
- The *type* determines if the protein is an *input* protein, the concentration of which is given by the environment of the GRN and which regulates other proteins but is not regulated, an *output* protein, the concentration of which is used as output of the network and which is regulated but does not regulate other proteins, or a *regulatory* protein, an internal protein that regulates and is regulated by other proteins.

The dynamics of the GRN is calculated as follow. First, the affinity of a protein a with another protein b is given by the enhancing factor u^+_{ab} and the inhibiting u^-_{ab}:

$$u^+_{ab} = p - |enh_a - id_b| \quad ; \quad u^-_{ab} = p - |inh_a - id_b| \tag{4}$$

where id_x is the tag, enh_x is the enhancer tag and inh_x is the inhibiting tag of protein x.

The GRN's dynamics are calculated by comparing the proteins two by two using the enhancing and the inhibiting matching factors. For each protein in the network, the global enhancing value is given by the following equation:

$$g_i = \frac{1}{N}\sum_{j}^{N} c_j e^{\beta u^+_{ij} - u^+_{max}} \quad ; \quad h_i = \frac{1}{N}\sum_{j}^{N} c_j e^{\beta u^-_{ij} - u^-_{max}} \tag{5}$$

where g_i (or h_i) is the enhancing (or inhibiting) value for a protein i, N is the number of proteins in the network, c_j is the concentration of protein j and u^+_{max} (or u^-_{max}) is the maximum enhancing (or inhibiting) matching factor observed. β is a control parameter described hereafter.

The final modification of protein i concentration is given by the following differential equation:

$$\frac{dc_i}{dt} = \frac{\delta(g_i - h_i)}{\Phi} \tag{6}$$

where Φ is a function that keeps the sum of all protein concentrations equal to 1.

β and δ are two constants that set up the speed of reaction of the regulatory network. In other words, they modify the dynamics of the network. β affects the importance of the matching factor and δ affects the level of production of the protein in the differential equation. The lower both values, the smoother the regulation. Similarly, the higher the values, the more sudden the regulation.

B. GRN-controlled neuromodulation

Neuromodulation is incorporated into RL-controlled robots according to the model shown in Figure 2. When evaluating the performance of a GRN, a robot is initialized with a uniform Q-function and the GRN is first run with no inputs for 50 steps. The aim of this initialization is to reach a stable point before exploiting the GRN with inputs. This phase is necessary because GRNs are known to oscillate chaotically in their very first steps.

After initialization, the robot is controlled according to the typical SARSA on-line policy learning mechanism; however, while the agent chooses an action, the proteins of the GRN react and tune the concentration of the neuromodulators. These neuromodulators are used for the learning and memory parameters, which are explained in sections III and VI. After observing the reward from taking an action in the current state, the SARSA update is applied as usual but with parameters determined by the GRN.

To regulate the RL parameters, the GRN uses the current state of the environment as inputs. In the CCP problem addressed in this paper, inputs correspond to the quantity of dirt in the agent's von Neumann neighborhood, the number of obstructed positions in the same neighborhood, and the current reward (or lack thereof). The GRN is then run for 5 steps with these inputs before the learning and memory parameters are used within the RL module. These 5 steps are necessary to reach a new stable state of the GRN with the new inputs.

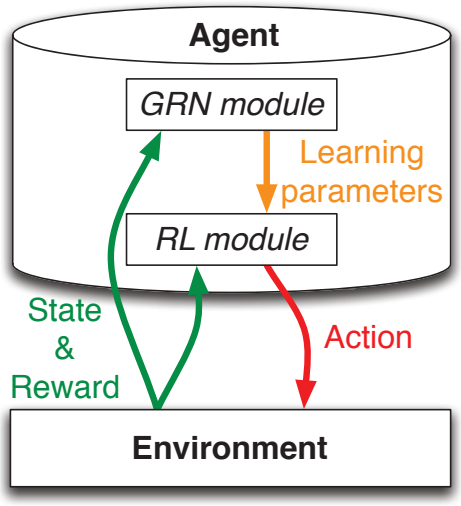


Fig. 2: A diagram of the model of neuromodulatory control of RL agents.

V. Experiment

In this study we show that the performance of RL-controlled robots on the CCP is exceeded by RL-controlled robots with neuromodulatory GRNs. The CCP has a natural framing of reward, which is the dirt that it cleans. The state of a robot is defined in terms of the robot's perceived surroundings. In this way, states are not unique with respect to the environment; many environments will be perceived as the same state. At each timestep the robot can choose between 4 actions of movement along the von Neumann neighborhood (North, East, South, West), and the world wraps as a torus. A robot receives a reward of 1 for moving to a dirty location, and otherwise receives no reward. These formulations of state, action, and reward are sufficient to allow RL to control the robot. A diagram of the experimental environment can be seen in Figure 1.

The primary complexity of the environments in our experiments stems from obstacles, which we expect only requires the learning of short sequences of actions. Vision extends for a radius of 2 in a cross about the von Neumann neighborhood. Each location in the discrete 10x10 training grid is represented with a ternary variable encoding: clean, dirty, or obstructed. The GRN perceives the environment through three signals: the current reward, fraction of visual field that is dirty, and the fraction of the visual field that is obstructed. Output proteins of the GRNs are used to specify the α, γ, and λ learning and memory parameters of the reinforcement learning algorithm, as well as a threshold protein used to

normalize the previously mentioned outputs. This integration of a neuromodulatory GRN and reinforcement learning algorithm is biologically plausible as well. In this way the GRN acts as a teacher for the robot, by guiding memory formation, and retention. Parameters are shown in Table I.

The evolutionary algorithm used to evolve the GRNs is similar to a (μ+λ)-evolution strategy, where subsequent generations are selected from both a population of parents and children. An initial population of randomly generated GRNs is created and evaluated. Half of the initial population is initialized with GRNs which have been filtered to ensure some stability in their output proteins. Stable networks pass 2 of 3 criteria which are applied to the parameter values produced by the network. Each critereon tests that the standard deviation of a given parameter has a standard deviation less than 0.25 with a mean centered at 0.75 for α, γ and 0.25 for λ. Populations are iteratively updated by creating a candidate child program for each parent, via mutation or crossover. The child is then evaluated and compared to its parent. If the child has a lower error than its parent, then it replaces its parent. The fitness used for evolution is the integral of error over all training episodes.

Generalization is tested by constructing a set of hypothetical test scenarios for 6 classes. These scenarios are states that the robot might otherwise encounter during training, but are only used to test the robot's behavior in a single state. By only considering a single state-action transition, it becomes tractable to derandomize the Markovian process and thus consider all possible outcomes. For a scenario state, s_{c1}, the agent may take any action $a_{c1} \in A_{c1}$. The generalization score is then

$$score_g = (1 - 3/4\epsilon) r(s_{c1}, \hat{a}_{c1}) + \epsilon/4 \sum_{a \in \hat{A}_{c1}} r(s_{c1}, a) \quad (7)$$

where $\hat{a}$ is the agent's preferred action and $\hat{A}_{c1}$ is all actions excluding $\hat{a}$. 5 of the 6 scenario classes correspond to 0-4 dirty squares surrounding the robot, and the 6th class is random. Scenarios are generated to contain typical distributions of dirt and obstacles encountered over the course of a training episode. The inclusion of these generalization scenarios allow the robot to be tested on its response to a wide range of environmental conditions that may not appear during a training episode.

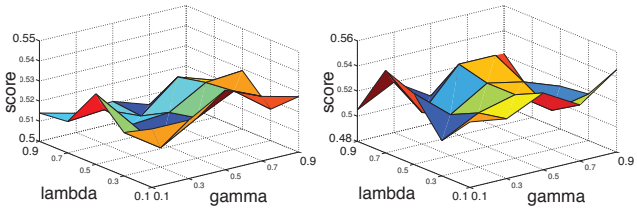
VI. Analysis of Parameters

In this analysis we investigate the core parameters of the temporal difference learning algorithm on the CCP problem for 0 obstacles or various maps with 30 obstacles. The main question we would like to answer is, how do the RL parameters affect the performance of learning agents on the CCP? In order to do this we perform a uniform sampling of the three parameters: α, γ, and λ from $[0.1, 1]$ with an interval of 0.1. The result of this sampling is a 4D space where the 3 parameter coordinates are matched with the fitness achieved by RL agents with these parameters. Agents are trained for 50 training episodes, where the environment is repopulated with dirt at the end of each episode. In order to display the parameter maps in a readable fashion we find the best parameter set from all sampling runs and focus on modal parameter values. While the subject parameters have highly non-linear effects on the behavior of RL agents, we articulate some of the general properties they can have.

The learning rate, α, controls how much the agent learns from the error of its Q-function relative to the observed reward. Large values of α can bias the agent towards behavioral feedback loops, by placing emphasis on rewards experienced early in training. The parameter maps for fixed values of α are shown in Figure 3. With a fixed value of alpha we see that there can be a wide range in behavior as we vary the values of γ and λ. The parametric performance surface suggests that there may be multiple optima with these fixed α values

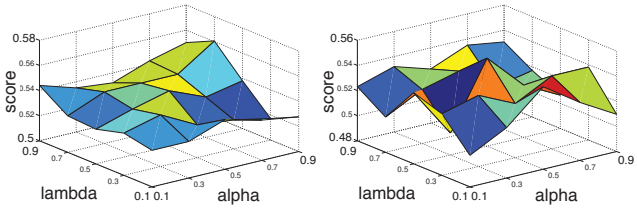
Parameter	Value
Sensor radius	2
World dimensions	10 x 10
ϵ, p(random move)	0.1
Training episodes	50
Steps per episode	200
Generalization cases	600 (6 classes)
Input proteins	3
Output proteins	4
GRN steps per action	5
Population size	50
Max. generations	50
Tournament size	7
Number of runs	25

TABLE I: Parameters used in experiments.



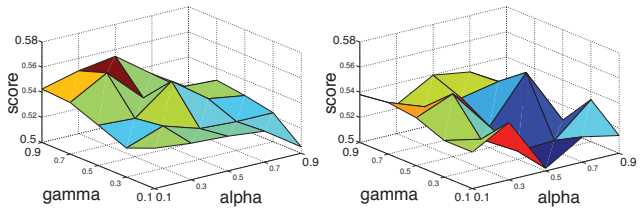
(a) 0 obstacles, $\alpha = 0.1$ (b) 30 obstacles, $\alpha = 0.7$

Fig. 3: Generalization score for the CCP trained with 0 obstacles with $\alpha = 0.1, 0.7$, the modal best alpha found during parameter analysis. The remaining RL parameters γ and λ are shown on the X- and Y-axes. Bigger score is better.



(a) 0 obstacles, $\gamma = 0.1$ (b) 30 obstacles, $\gamma = 0.3$

Fig. 4: Generalization score for the CCP trained with 0 obstacles with $\gamma = 0.1, 0.3$, the modal best gamma found during parameter analysis. The remaining RL parameters α and λ are shown on the X- and Y-axes. Bigger score is better.



(a) 0 obstacles, $\lambda = 0.7$ (b) 30 obstacles, $\lambda = 0.5$

Fig. 5: Generalization score for the CCP trained with 0, 30 obstacles with $\lambda = 0.7, 0.5$, the modal best lambda found during parameter analysis. The remaining RL parameters α and γ are shown on the X- and Y-axes. Bigger score is better.

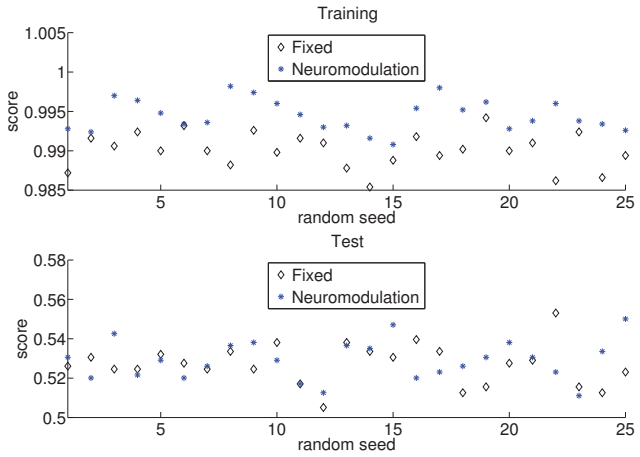


Fig. 6: Training (top) and generalization (bottom) scores for RL and neuromodulation per random seed for 0 obstacles.

for both the 0 and 30 obstacles cases. We will return to this observation after examining the remaining parameters.

The discounting factor γ encodes the discounting factor of future rewards such that a reward 5 timesteps in the future is discounted by a factor of γ^4. Agents with low values of γ are relatively short sighted, favoring actions that return larger immediate rewards at the cost of longer-term considerations. The parameter maps for fixed values of γ are shown in Figure 4. As we keep the values of γ fixed we see that for 0 obstacles large values of α and λ lead to the best agent behavior. On the other hand, in the case of 30 obstacles the best values of α and λ are unclear, with most pairings performing equivalently.

The trace decay λ is used to discount the significance of actions leading up to a reward. Larger values of λ imply that the reward an agent receives was highly dependent on a long sequence of actions that led to the reward. The parameter maps for fixed values of λ are shown in Figure 5. Again, we see different behavior for 0 and 30 obstacles. In the case of 0 obstacles it is fairly clear that large values of γ and small values of α are best. However, in the case of 30 obstacle maps the performance of parameters is again noisier. The highest peak is for $\alpha = 0.7$ and $\gamma = 0.5$. The non-uniformity that is observed over the distribution of parameters leads us to consider that the idea of choosing an optimal set of

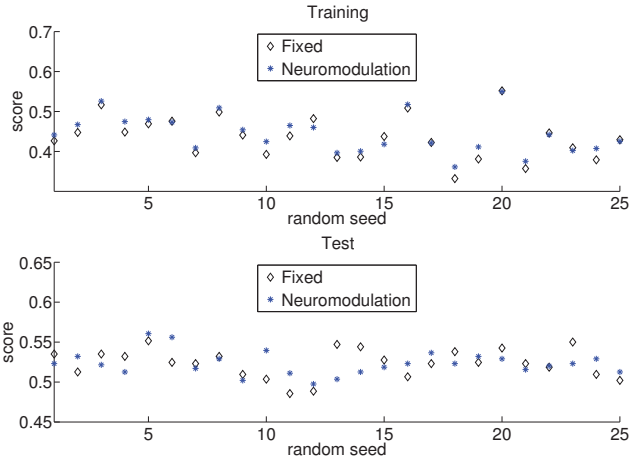


Fig. 7: Training (top) and generalization (bottom) scores for RL and GRN per random seed for 30 obstacles.

parameters is not only non-trivial, but may not be possible.

We see that parameter space can contain multiple optima and becomes more complex as environmental complexity is increased. One might suspect that these multiple optima correspond to different types of behavior that are learned by the RL agent. Regardless of how the optima manifest in terms of behavior, the possibility of dynamically adjusting the learning parameters to the RL agent may be a way of harnessing the some of the properties of RL parameters over the course of a series of training episodes. In the following section we present results for the dynamic control of the learning parameters by a GRN.

VII. Evolved Neuromodulation

In these experiments we present a comparison of the performance of uniformly sampled RL parameters with the performance of GRN-controlled neuromodulation of the same RL parameters. The dynamic control of learning and memory parameters by this neuromodulatory process provides the agent with the means for adaptive control of its own learning process. We show that agents who have been trained with neuromodulatory GRNs controlling their learning and memory parameters, outperform RL agents that have been trained with fixed parameters.

In Table II, we see a comparison of RL agents that have been trained with the best fixed parameters found with uniform sampling, and RL agents that have been trained with neuromodulatory GRNs. Best fixed parameters are a tuple of $(\alpha, \gamma, \lambda)$ that produce the highest score at the end of training. Performance is compared on the training map, as well as on a set of hypothetical scenarios designed to test generalization. In the experiments with 0 obstacles, neuromodulatory GRNs outperform fixed parameters in both training and test scores. The CCP with 0 obstacles is a relatively easy problem that can be solved with simple patterns such as row-by-row cleaning. The improved performance with neuromodulation suggests that the evolved GRN is guiding the reinforcement

# Obstacles	Type	$score_{training}$	$score_{test}$
0	RL	0.99006(stdev 0.0022557)	0.50496(stdev 0.0041289)
0	GRN	0.9945(stdev 0.0020109)	0.52914(stdev 0.010029)
30	RL	0.43441(stdev 0.052902)	0.49056(stdev 0.008722)
30	GRN	0.44452(stdev 0.047907)	0.52326(stdev 0.014845)

TABLE II: Results for performance of RL with the best sampled parameters and evolved GRN-controlled learning. $score_{training}$ represents the fraction of dirt cleaned over all training episodes (including initially naive behavior). $score_{test}$ represents the dirt that is cleaned during a set of hypothetical scenarios. In both cases bigger is better.

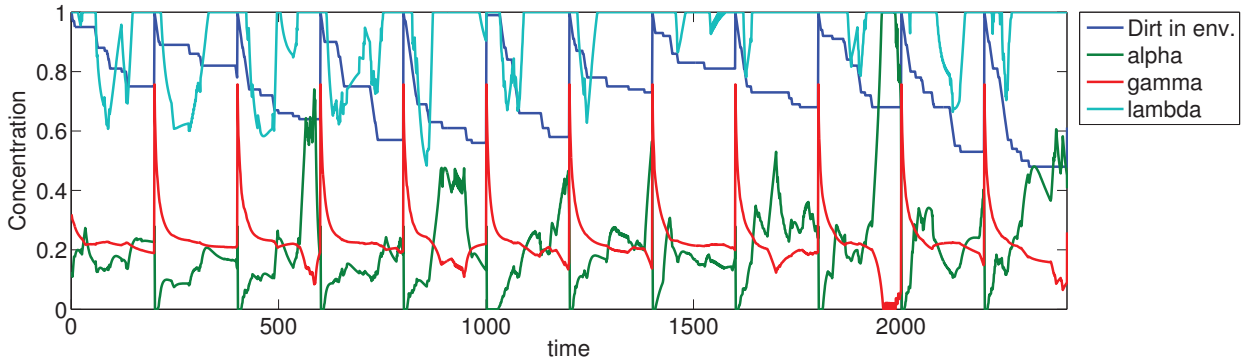


Fig. 8: Example of neuromodulation of learning and memory parameters α, γ, and λ by an evolved neuromodulatory GRN. This GRN is the best-of-run for a 30 obstacle map. These are the first 12 training episodes, where spikes at intervals of 200 indicate a resetting of the environment.

learning algorithm in meaningful ways in response to environmenal cues. This is particularly visible in Figure 6 where neuromodulation is always better than pure RL during training and for most tests.

In experiments with 30 obstacles the results are not as clear-cut as in the case of 0 obstacles. Figure 7 details these result per environment seed. Although the mean and standard deviation of training score for neuromodulatory GRNs is slightly lower than those of fixed parameter RL, it is not significant with respect to the standard deviation. However, agents trained with neuromodulatory GRNs perform better on the set of hypothetical test scenarios. This suggests that GRNs are not only guiding the RL algorithm in meaningful ways, but that the GRN is helping the RL agent to learn generalizable behaviors.

An example of one of the best-of-run GRNs from a 30 obstacle map is shown in Figure 8. This figure shows an example training session with a neuromodulatory GRN. While fixed parameter RL experiments maintain constant values for α, γ, and λ, the neuromodulatory GRN dynamically modulates these learning and memory parameters. This particular GRN uses a strategy of a high γ value at the beginning of each training episode, which decays over the course of the episode. λ values are maintained near unity, which will heavily reinforce actions that have indirectly contributed to receiving a reward. Of particular interest is the increase of α in response to decreases in γ. This dynamic will cause agents to initially populate their Q-function relatively quickly, and then will lead them to learn from their immediate rewards. This behavior can be particularly significant after most of the dirt has been removed from the environment. Systematic errors which repeatedly leave dirt in similar situations, such as those hard to reach corners that continually accumulate dust, can be learned by these agents which begin to learn more from immediate rewards.

VIII. Discussion and Conclusions

In this study we have introduced the evolution of neuromodulatory GRNs into the framework of reinforcement learning by modulating both learning and memory during on-line learning. Evolved neuromodulatory GRNs train RL agents that consistently have improved generalization abilities relative to RL agents trained with fixed parameter sets on the coverage control problem. The GRNs dynamically adjust learning parameters relative to environmental inputs, this dynamic adjustment allows agents to regulate changes in the behavior by altering how behaviors are learned.

The Baldwin effect is a process in which learning allows an individual to probe possible strategies. Successful strategies can then be evolved and incorporated into the genome by selective pressures. However, this study approaches the Baldwin effect from a different perspective than those of Hinton and Nowlan where learning and evolution act on the same substrate, such as the weights of a neural network. In our case an evolved neuromodulatory GRN purely controls the learning process. This is distinct from the Hinton and Nowlan class of experiments where learning, in a sense, preceeds evolution which eventually learns fixed encodings of information that would otherwise be learned.

RL agents that are trained with neuromodulatory GRNs

can outperform fixed parameter RL agents in training environments in some cases; however, these GRNs consistently train agents that have improved generalization capabilities. In this sense, the neuromodulatory GRN serves as an excellent teacher to the RL agent, by guiding the learning process in response to environmental variation. This may allow neuromodulated RL agents to utilize multiple phases of learning, whereby it is possible to learn how to correct previous mistakes. These reasons suggest that it may be interesting to consider neuromodulatory GRNs within dynamic environments, where learning on-the-fly can be critical.

Future work should investigate the effect of neuromodulatory GRNs in dynamic environments, as well as in long-term evolution experiments. Extensions to include additional neuromodulators, like those of [14], may further improve performance. The enhanced generalization abilities of agents trained with neuromodulatory GRNs suggests that this framework may be useful in highly complex environments. Therefore, testing neuromodulatory GRNs in other foraging tasks, such as hunting with high-dimensional visual input [28], may prove fruitful. Ultimately, neuromodulatory GRNs should be studied in neural substrates of higher-order complexity.

IX. Acknowledgements

We thank Yasmin Escobedo Lozoya, Jessica Lowell, Hava Siegelmann, and Richard Watson for discussions that contributed to this work. We also thank Francesco Pontiggia and the Brandeis HPC for providing computational support.

References

[1] D Ackley and M Littman. Interactions between learning and evolution. In *Artificial life II*, pages 487–509, 1991.

[2] L. W. Ancel. Undermining the Baldwin expediting effect: Does phenotypic plasticity accelerate evolution? *Theoretical population biology*, 58(4):307–319, 2000.

[3] LW Ancel and W Fontana. Plasticity, evolvability, and modularity in RNA. *Journal of Experimental Zoology*, 288(3):242–283, 2000.

[4] JM Baldwin. A new factor in evolution. *American naturalist*, 30(354):441–451, 1896.

[5] W. Banzhaf. Artificial Regulatory Networks and Genetic Programming. In Rick L Riolo and Bill Worzel, editors, *Genetic Programming Theory and Practice*, chapter 4, pages 43–62. 2003.

[6] L Benuskova and N Kasabov. Modeling brain dynamics using computational neurogenetic approach. *Cognitive neurodynamics*, 2(4):319–334, 2008.

[7] H Choset. Coverage for robotics: A survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1):113–126, 2001.

[8] J Cortes and S Martinez. Coverage control for mobile sensing networks. *Robotics and Automation, IEEE Transactions on*, 20(2):243–255, 2004.

[9] S Cussat-Blanc, J Pascalie, S Mazac, H Luga, and Y Duthen. A synthesis of the Cell2Organ developmental model. *Morphogenetic Engineering*, (353-381), 2012.

[10] S Cussat-Blanc, S Sanchez, and Y Duthen. Controlling Cooperative and Conflicting Continuous Actions with a Gene Regulatory Network. In *Conference on Computational Intelligence in Games (CIG'12)*, pages 187–194. IEEE, 2012.

[11] A Destexhe and E Marder. Plasticity in single neuron and circuit computations. *Nature*, 431(7010):789–795, 2004.

[12] D. Dickmanns, J. Schmidhuber, and A. Winklhofer. Der genetische algorithmus: Eine implementierung in prolog. *Fortgeschrittenenpraktikum, Institut f ur Informatik, Lehrstuhl Prof. Radig, Technische Universit at M unchen*, 1987.

[13] R Doursat. Facilitating evolutionary innovation by developmental modularity and variability. In *Proceedings of the 11th annual conference on genetic and evolutionary computation*, pages 683–690, 2009.

[14] K Doya. Metalearning and neuromodulation. *Neural Networks*, 15(4):495–506, 2002.

[15] JM Fellous and C Linster. Computational models of neuromodulation. *Neural computation*, 10(4):771–805, 1998.

[16] SN Haber and B Knutson. The reward circuit: linking primate anatomy and human imaging. *Neuropsychopharmacology*, 35(1):4–26, 2009.

[17] G.E. Hinton and S.J. Nowlan. How learning can guide evolution. *Complex systems*, 1(1):495–502, 1987.

[18] M Joachimczak and B Wróbel. Evo-devo in silico: a model of a gene network regulating multicellular development in 3D space with artificial physics. In *Proceedings of the 11th International Conference on Artificial Life*, pages 297–304, 2008.

[19] M Joachimczak and B Wróbel. Evolving Gene Regulatory Networks for Real Time Control of Foraging Behaviours. In *Proceedings of the 12th International Conference on Artificial Life*, pages 348–355, 2010.

[20] PS Katz. Intrinsic and extrinsic neuromodulation of motor circuits. *Current opinion in neurobiology*, 5(6):799–808, 1995.

[21] J.R. R Koza. *Genetic programming II: automatic discovery of reusable programs*, volume 1. 1994.

[22] JL Krichmar. The neuromodulatory system: a framework for survival and adaptive behavior in a challenging world. *Adaptive Behavior*, 16(6):385–399, 2008.

[23] E Marder. Neuromodulation of neuronal circuits: back to the future. *Neuron*, 76(1):1–11, 2012.

[24] E Marder and V Thirumalai. Cellular, synaptic and network effects of neuromodulation. *Neural Networks*, 15(4):479–493, 2002.

[25] P.R. Montague, P. Dayan, and T.J. Sejnowski. A framework for mesencephalic dopamine systems based on predictive Hebbian learning. *The Journal of Neuroscience*, 16(5):1936–1947, 1996.

[26] R Morlok and M Gini. Dispersing robots in an unknown environment. In *Distributed Autonomous Robotic Systems 6*, pages 253–262, 2007.

[27] M. Nicolau, M. Schoenauer, and W. Banzhaf. Evolving Genes to Balance a Pole. In Anna Isabel Esparcia-Alcazar, Aniko Ekart, Sara Silva, Stephen Dignum, and A Sima Uyar, editors, *Proceedings of the 13th European Conference on Genetic Programming, EuroGP 2010*, volume 6021 of *LNCS*, pages 196–207, 2010.

[28] ME Palmer and A Chou. Evolved neural network controllers for physically simulated robots that hunt with an artificial visual cortex. In *Artificial Life*, pages 415–422, 2012.

[29] L Pitonakova. Ultrastable neuroendocrine robot controller. *Adaptive Behavior*, 21(1):47–63, 2013.

[30] JB Pollack. Cascaded back propagation on dynamic connectionist networks. In *Proceedings of the Ninth Conference of the Cognitive Science Society*, pages 391–404, 1987.

[31] S Risi, SD Vanderbleek, CE Hughes, and KO Stanley. How novelty search escapes the deceptive trap of learning to learn. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference:*, pages 153–160, 2009.

[32] SJ Russell, P Norvig, JF Canny, JM Malik, and DD Edwards. *Artificial intelligence: a modern approach*. 1995.

[33] W Schultz, P Apicella, and T Ljungberg. Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task. *The Journal of Neuroscience*, 13(3):900–913, 1993.

[34] A Soltoggio, JA Bullinaria, and C Mattiussi. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. *Artificial Life*, 11:569–576, 2008.

[35] L. Spector, K. Harrington, B. Martin, and T. Helmuth. What's in an evolved name? The evolution of modularity via tag-based reference. In *Genetic Programming Theory and Practice*, pages 1–16, 2011.

[36] O Sporns and WH Alexander. Neuromodulation and plasticity in an autonomous robot. *Neural Networks*, 15(4):761–774, 2002.

[37] RS Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[38] RS Sutton and AG Barto. *Reinforcement learning: An introduction*. 1998.

[39] A Wagner. Does evolutionary plasticity evolve? *Evolution*, 50(3):1008–1023, 1996.

[40] Y Zhurov and V Brezina. Variability of motor neuron spike timing maintains and shapes contractions of the accessory radula closer muscle of Aplysia. *The Journal of neuroscience*, 26(26):7056–7070, 2006.

[41] J. Ziegler and Wolfgang Banzhaf. Evolving Control Metabolisms for a Robot. *Artificial Life*, 7(2):171–190, 2001.